Information granulation in automated modeling

Matthew Easley
Elizabeth Bradley

CU-CS-905-00

Information granulation in automated modeling

Matthew Easley and Elizabeth Bradley

Department of Computer Science, University of Colorado, Boulder, CO 80309-0430, USA

Abstract. The goal of input-output modeling is to apply a test input to a system, analyze the results, and learn something useful from the cause-effect pair. Any automated modeling tool that takes this approach must be able to reason effectively about sensors and actuators and their interactions with the target system. The granulation level of the information involved in this process ranges from low-level data analysis techniques to abstract, qualitative observations about the system. This chapter describes a knowledge representation and reasoning framework that allows this process to be automated.

Keywords. Automated model building, system identification, input-output modeling, knowledge representation

Technical Report CU-CS 905-00

To appear in *Granular Computing: An Emerging Paradigm*

1 Input-Output Modeling

One of the most powerful analysis and design tools in existence—and often one of the most difficult to create—is a good model. Modeling is an essential first step in a variety of engineering and scientific problems. Faced with the task of designing a controller for a robot arm, for instance, a mechanical engineer performs a few simple experiments on the system, observes the resulting behavior, makes some informed guesses about what model fragments could account for that behavior, and then combines those terms into a model and checks it against the physical system. The information involved in this process is heterogeneous both in type and in level. The observations of the target system, for instance, can range from detailed sensor data to abstract, qualitative information like "the temperature oscillates." In order to create a model that is both useful (i.e., that captures the desired behavior) and minimal (i.e., that does not contain extraneous modeling components), engineers must granulate that information in appropriate ways. The topic of this chapter is a knowledge representation and reasoning framework that allows this process to be automated.

Modeling is an extremely broad research area, with roots—and applications—in fields ranging from artificial intelligence and cognitive psychology to control theory and engineering. We do not attempt a comprehensive survey of the uses of granulation across all of these fields; rather, we concentrate only upon the use of granular information in input-output modeling: that is, in the phase of the model-building process that is concerned with physical observations of the target system. We further restrict our attention to deterministic dynamical systems—in particular, those that can be modeled with ordinary differential equations (ODEs).

The goal of input-output modeling of dynamical systems is to apply a test input to the system, analyze the results, and learn something useful from the cause/effect pair. Reasoning about this procedure at a low level of granularity is tedious and difficult. Raw sensor data, for instance, is often both excessive and lacking: one has megabytes of noisy measurements, but of only one of the system's many state variables. Actuator interactions are even harder, since any reasoning about actuators must factor in the interface between the actuator and the system—a difficult modeling problem in its own right, and one that is all but impossible if one must solve it by manipulating voltage values and waveform timing.

An engineer's fundamental formalized knowledge lets him or her solve these problems by reasoning about sensors and actuators at a variety of levels, depending on the requirements of the problem at hand. Determining something as simple as "the voltage oscillates between 5 and 10 volts," for example, can be difficult if one attempts to scan an ASCII text file, but it is trivial to see when presented on an oscilloscope. This type of granular knowledge is critical in the model-building process, because qualitative observations play a much more wide-ranging role than a highly situation-specific sensor data set. Granulation is equally powerful in actuator-related reasoning. Any non-trivial dynamical system has multiple behavioral regimes, and identifying and characterizing these regimes is extremely useful (e.g., the *bifurcation diagram* that is commonly used to describe a dynamical system). The goal of this chapter is to show how automated tools can capture this kind of reasoning, generating and using high-level, granular knowledge that is useful to the model building process. The following two sections describe tools that use granular techniques to solve the sensor data analysis problem and the actuator control problem, respectively; we then close with a brief discussion of related work.

2 Granulating Sensor Data

Efficient model building requires the distillation of succinct, meaningful conclusions about a complicated system out of reams of sensor data. An effective way to do this is to apply geometric reasoning techniques to the data, as described in the following section. Many geometric reasoning techniques, however, require a full state-space trajectory, and fully *observable* systems are rare in engineering practice. Often, some of the state variables are either physically inaccessible or cannot be measured with available sensors. If the target system has 34 state variables, for example, and one can only measure one of those 34 signals, it would appear that the conclusions that one can draw from the sensor data are fundamentally limited. This is control theory's *observer problem*: the task of inferring the internal state of a system solely from observations of its outputs. Delay-coordinate embedding, a good solution to this problem, creates an m-dimensional *reconstruction-space* vector from m time-delayed samples of data from a single sensor; see Section 2.2. The central idea is that the reconstruction-space dynamics and the true (unobserved) state-space dynamics are topologically identical, which implies that a state-space portrait reconstructed from a single sensor time series is qualitatively identical to the true multidimensional dynamics of the system. Together, delay-coordinate embedding and geometric reasoning techniques allow effective granulation of sensor data for automated model building.

2.1 Distilling Qualitative Information from Quantitative Data

A variety of techniques have been developed for extracting qualitative properties from a numeric data set. The solution described here combines phase-portrait analysis, asymptote recognition, and computer-vision techniques. Dynamical systems practitioners typically reason about phase portraits, rather than time series, because the phase-portrait representation—which suppresses time and plots only the state variables—brings out the qualitative properties of the system under examination in a very natural way. For example, recognizing a damped oscillation in a time series from a linear system requires detailed examination of the amplitude decay rate of and the phase shift between two decaying sinusoidal time-domain signals. The same behavior manifests in a much more obvious form—a single symmetric spiral—on a phase portrait. Automated phase-portrait analysis techniques[3,24,25] are designed to capture this kind of information and generate the corresponding qualitative descriptions. This kind of granular information is perfectly suited for automated model building; its abstract, broadly applicable nature allows the automatic verification or rejection of large classes of candidate models. For example, if sensor measurements of a state variable indicate that it is undergoing a damped oscillation, one can immediately rule out all linear ODE models that are unstable, critically damped, or overdamped[1].

One of the earliest and most powerful phase-portrait analysis techniques is the cell dynamics formalism of Hsu[13,14], which discretizes a set of n-dimensional state vectors onto an n-dimensional mesh of uniform boxes or *cells*. In Figure 1(a), for example, the circular trajectory—a sequence of two-vectors of floating-point numbers measured by a finite-precision sensor—can be represented as the *cell sequence*

$$[...(0,0)(1,0)(2,0)(3,0)(4,0)(4,1)(4,2)(4,3)(4,4)(3,4)...]$$

Because multiple trajectory points are mapped into each cell, this dis-

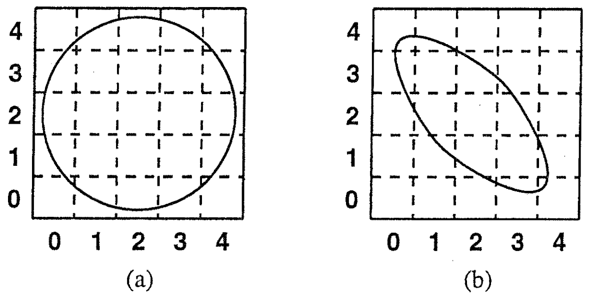

FIGURE 1. Identifying a limit cycle using simple cell mapping. After [4].

cretized representation of the dynamics is significantly more compact than the original series of floating-point numbers and therefore much easier to work with. This is particularly important when complex systems are involved, as the number of cells in the grid grows exponentially with the number of dimensions[2]. Although the approximate nature of the cell dynamics representation does abstract away much detailed information about the dynamics, it preserves many of its important invariant properties. This point is crucial to the fidelity of this analysis method; it means that conclusions drawn from the discretized trajectory are also true of the real trajectory—e.g.,, a repeating sequence of cells implies that the true dynamics are also on a limit cycle. In this manner, low-level, finite-precision numerical data can be converted into a high-level qualitative classification. Its coarse-grained nature confers some important limitations upon this scheme—both subtle and obvious—many of which are described below; see [9] for more details. Another key concept of the cell dynamics formalism is that it allows a

[1] Stability analysis of a linear ODE involves finding the roots of its *characteristic polynomial* (e.g., $as^2 + bs + c = 0$ for the ODE $a\ddot{x} + b\dot{x} + c = 0$). Only if those roots are imaginary can the system oscillate; only if their real parts are negative is the oscillation damped.

[2] The example of Figure 1 is two-dimensional, but the cell dynamics formalism generalizes smoothly to arbitrary dimension.

set of geometrically different and yet qualitatively similar trajectories—an "equivalence class" with respect to some important dynamical property—to be classified as a single coherent group of phase portraits. Part (b) of Figure 1 shows a different trajectory with identical topology; this, too, would be classified in the limit cycle equivalence class by the cell dynamics algorithm. See, e.g., Hao[11] or Lind & Marcus[17] for more details.

Given the cell dynamics formalism described in the previous paragraph, the dynamics of a discretized trajectory can be quickly and qualitatively classified using simple geometric heuristics. Some of these classification heuristics are trivial (e.g. determining if the trajectory exits the mesh), but detecting limit cycles or oscillations requires subtler pattern recognition techniques. Below are several of our geometric reasoner's dynamics classifications, the corresponding heuristics, and some associated implications for the ODE model:

- `fixed-cell`: when a trajectory relaxes to a single cell and remains within that cell for a fixed percentage of its total lifetime. This can, for instance, be used to recognize when a system is damped. Appropriate mesh geometry choices can extend this method to asymptote recognition.
- `limit-cycle`: when the trajectory contains a finite, repeating sequence of cells. These patterns are identified by discarding any transients and searching for periodic mapping sequences; they indicate that the system is either conservative or externally driven.
- `damped-oscillation`: when a trajectory enters a fixed cell via a decaying oscillation. This pattern is detected by recognition of an inward spiral; such dynamics can indicate, for instance, that a linear system is underdamped and thus that at least one pair of the model's natural frequencies must be complex.
- `constant`: when a state variable does not change over the duration of the trajectory. This computation involves a simple serial scan on each mesh axis; its results are particularly useful in the model-building process because they have wide-ranging implications about the order of the system.
- `sink-cell`: when a trajectory exits the mesh. This information is used to identify unstable trajectories.

Many other classifications are possible (e.g., that the system is chaotic); some are less useful than others—because their implications either are more limited in range or require processing at a less-abstract reasoning level.

The cell size, mesh boundary, and trajectory length affect the validity and efficiency of the cell dynamics classification. Among other things, a small limit cycle—one that is contained within a single cell—may be classified as a fixed point, and behavior outside the mesh will not be classified at all. All of these discretization and boundary effects are not, in fact, problems; rather, they actually allow one to explicitly represent and work with the abstraction levels implied by the finite range and resolution that are such fundamental features of a modeling hierarchy—e.g., to avoid including saturation and crossover distortion effects when asked to "model the *small-signal* behavior of the op amp *to within 10mV*." Specifically, the range and resolution information may be used to set up the mesh boundary and cell size, assuring that the behavior outside the range or below the resolution is not modeled.

2.2 Delay-Coordinate Methods for Observer Theory

If all of a system's state variables are identified and measured, the geometric reasoning techniques described in Section 2.1 can be applied directly to the sensor data. A fully *observable* system like this, however, is rare in engineering practice; as a rule, many—often, most—of the state variables either are physically inaccessible or cannot be measured with available sensors. Worse yet, the true state variables may not be known to the user; temperature, for instance, can play an important and often unanticipated role in the modeling of an electronic circuit. This is, as mentioned briefly before, part of control theory's observer problem: how to (1) identify the internal state variables of a system and (2) infer their values from the signals that *can* be observed. The arsenal of time-series analysis methods developed by the nonlinear dynamics community in the past decade[1] provides powerful solutions to both parts of this problem. This section describes two methods, Pineda-Sommerer (P-S)[21] and false near neighbor (FNN)[16], that may be used to infer the dimension of the internal system dynamics from a time series measured by a single output sensor[3].

Both P-S and FNN are based on *delay-coordinate embedding*, wherein one constructs m-dimensional *reconstruction-space* vectors from m time-delayed samples of the sensor data. For example, if the time series in Figure 2 is embedded in three dimensions ($m = 3$) with a delay of 0.2, the first two points in the reconstruction-space trajectory are (32.0 22.0 19.0) and (28.0 16.0 23.0). Sampling a single system state variable is equivalent to projecting a d-dimensional state-space dynamics down onto one axis; embedding is akin to "unfolding" such a projection, albeit on different axes. Consider the classic Lorenz attractor, the first recognized instance of chaotic behavior[18], shown in Figure 3(a). Part (b) of the figure shows the results of sampling only the x coordinate of that three-dimensional trajectory and plotting it versus time—exactly the situation that would arise if one only had access to a single sensor. The embedded version of this one-dimensional time series, shown in part (c), is slightly distorted but qualita-

[3] Techniques like divided differences can, in theory, be used to derive velocities from position data; in practice, however, these methods often fail because the associated arithmetic magnifies sensor error.

t	x
0.1	32.0
0.2	28.0
0.3	22.0
0.4	16.0
0.5	19.0
0.6	23.0

$\vec{r}(0.1) = (32.0\ 22.0\ 19.0)$

$\vec{r}(0.2) = (28.0\ 16.0\ 23.0)$

FIGURE 2. An example delay-coordinate embedding with an embedding dimension of three and a delay of 0.2. x is the measured state variable; r is the vector in reconstruction space.

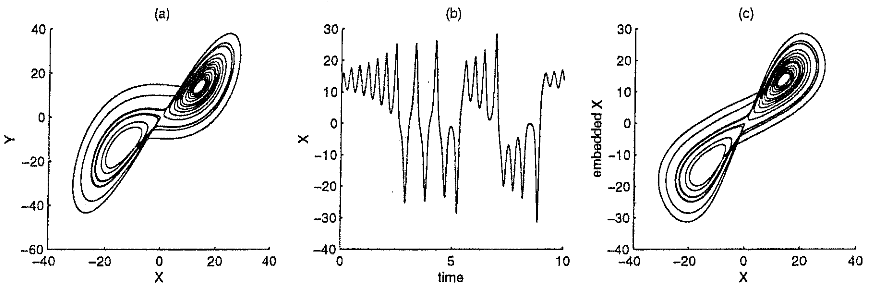

FIGURE 3. The Lorenz Attractor: (a) true version (x-y) (b) time series of x-axis (t-x) (c) embedded version (x-x').

tively identical to part (a). The central theorem relating such embeddings to the true, underlying dynamics was suggested in [23] and proved in [20]; informally, it states that given enough dimensions (m) and the right delay (τ), the reconstruction-space dynamics and the true (unobserved) state-space dynamics are topologically identical[4]. This is an extremely powerful correspondence: it allows one to analyze the underlying dynamics using only the output of a single sensor. In particular, many properties of the dynamics, such as dimension (i.e., whether the trajectory is a fixed point, limit cycle, chaotic attractor, etc.), are preserved by diffeomorphisms; if they are present in the embedding, they exist in the underlying dynamics as well. There are, of course, some important caveats, and the difficulties that they pose are the source of most of the effort and subtlety in these types of methods. Specifically, in order to embed a data set, one needs m and τ, and neither of these parameters can be measured or derived from

[4] More formally, the reconstruction-space and state-space trajectories are diffeomorphic iff $m \geq 2d + 1$, where d is the true dimension of the system.

the data set, either directly or indirectly, so algorithms typically rely on numeric and geometric heuristics to estimate them.

The Pineda-Sommerer algorithm creates such estimates; it takes a time series and returns the delay τ and a variety of different estimates of the dimension m. The procedure has three major steps: it estimates τ using the mutual information function, uses that estimated value τ_0 to compute a temporary embedding dimension E, and uses E and τ_0 to compute the *generalized dimensions* D_q, also known as "fractal dimensions." The standard algorithm for computing the fractal dimension of a trajectory, loosely described, is to discretize state space into ϵ-boxes, count the number of boxes occupied by the trajectory, and let $\epsilon \to 0$. Generalized dimensions are defined as

$$D_q = \frac{1}{q-1} \limsup_{\epsilon \to 0} \frac{\log \sum_i p_i^q}{\log \epsilon} \tag{1}$$

where p_i is some measure of the trajectory on box i. D_0, D_1, and D_2 are known, respectively, as the capacity, information, and correlation dimensions; all three are useful as estimates of the number of state variables in the system. The actual details of the P-S algorithm are quite involved; we will only present a qualitative description:

- Construct 1- and 2-embeddings of the data for a range of τs and compute the saturation dimension D_* of each; the first minimum in this function is τ_0. The D_* computation entails:
  - Computing the information dimension D_1 for a range of embedding dimensions E and identifying the saturation point of this curve, which occurs at D_*. The D_1 computation entails:
    - Embedding the data in E-dimensional space, dividing that space into E-cubes that are ϵ on a side, and computing D_1 using equation (1) with $q = 1$.

Ideally, of course, one lets $\epsilon \to 0$ in the third step, but floating-point arithmetic and computational complexity place obvious limits on this; instead, one repeats the calculation for a range of ϵs and finds the power-law asymptote in the middle of the log-log plot of dimension versus ϵ. P-S incorporates an ingenious complexity-reduction technique: the ϵs are chosen to be of the form 2^{-k} for integers k and the data are integerized; this allows most of the mathematical operations to proceed at the bit level and vastly accelerates the algorithm. To increase the precision of this computation, we have implemented an arbitrary-length virtual integer package that facilitates the integerization.

The false near neighbor algorithm is far simpler than P-S. It takes a τ and a time series[5] and returns m. FNN is based on the observation that

[5] P-S may be used to generate τ for use in FNN. Other methods, such as autocorrelation[1] can also be used to estimate τ.

neighboring points may in reality be projections of points that are very far apart, as shown in Figure 4. The algorithm starts with $m = 1$, finds each

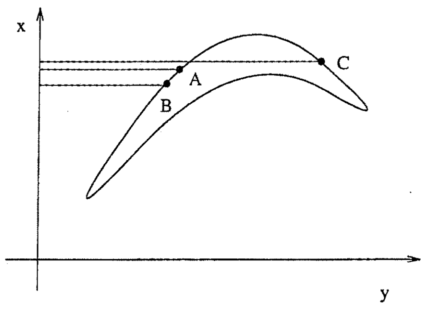

FIGURE 4. The geometric basis of the FNN algorithm: the points labeled A and B are true near neighbors in the x-projection, while A and C are false near neighbors. After [4].

point's nearest neighbor, and then re-embeds the data with $m = 2$. If the point separations change abruptly between the 1- and 2-embeddings, then the points were *false* neighbors (like A and C in the x-projection of Figure 4). The FNN algorithm continues adding dimensions until an acceptably small[6] number of false near neighbors remain, and returns the last m-value as the estimated dimension. A K-D tree implementation[10] reduces the complexity of the nearest-neighbor step from $O(N^2)$ to $O(N \log N)$, where N is the length of the time series.

As both FNN and P-S are based on heuristics, their estimates of the embedding dimension m are not necessarily the same. Since both algorithms provide conservative estimates, choosing the minimum of the two results gives a reasonable lower bound for the dimension of the model. This knowledge facilitates granulation of the reasoning involved in the model-building process, as it allows the modeler to rule out entire branches of the search space of possible models (e.g., all models whose dimension is below that lower bound).

3 Granulating Actuator Control

Analysis of sensor data is only a small part of understanding the entire system, since dynamic systems—by their very nature—are not passive objects. Rather, they have inputs and outputs, and the relationship between the two is a critical feature of the system's behavior, and thus an important part of its model. Moreover, any non-trivial dynamical system has multiple behavioral regimes, and any successful model builder must be able to

[6] An algorithm that removes *all* false near neighbors can be unduly sensitive to noise.

reason about this property. This can be a daunting task, even for human experts; selecting and exploring appropriate ranges of state variables, parameter values, etc., is a subtle and difficult problem that has received much attention in the qualitative reasoning community[2,3,24,25]. Manipulation of the available actuators and sensors so as to actually carry out a specific experiment is another difficult problem. Fortunately, granular information can help engineers and scientists manipulate a system's inputs *and* observe its outputs, exploring its different operating regimes without generating overwhelming amounts of data.

This section describes a knowledge representation and reasoning framework called *qualitative bifurcation analysis* or QBA which allows a computer to mimic the kind of analysis an engineer would perform in the input-output analysis of an unknown system. This framework, which is designed to support reasoning about the effects of control parameters and the existence of multiple behavioral regimes, is based on ideas from hybrid systems, nonlinear dynamics, and computer vision. Its representation is a hybrid construct termed the *qualitative state/parameter (QS/P)* space, which combines information about the behavior of a complex system and the effects of its control parameters (inputs) upon its behavior. QBA's reasoning procedures emulate a classic technique in the dynamical systems literature known as *bifurcation analysis*, wherein a human expert changes a control parameter, classifies the resulting behavior, determines the regime boundaries, and groups similar behaviors into equivalence classes. Putting QBA's ideas into physical practice requires yet another layer of granulation, which translates abstract concepts about experiments, such as "measure the step response," into low-level commands that manipulate actuators and sensors in appropriate ways.

Working together, QBA's representation and reasoning processes allow the automatic generation of the kind of observations that a human engineer would make about the system, such as

> "in the temperature range from 0 to $50°C$, the system undergoes a damped oscillation to a fixed point at $(x, y) = (1.4, -8)$; when $T > 50°C$, it follows a period-two limit cycle located at..."

As described before, this kind of granulated information is useful in that it raises the abstraction level of the model-building process.

3.1 Qualitative Bifurcation Analysis: the Representation

As described in Section 2.1, a discretized version of the state-space representation can effectively abstract away many of the low-level details about the dynamics of a system while preserving its important qualitative and invariant properties. Using the cell dynamics representation, in particular, the dynamics of a trajectory can be quickly and qualitatively classified using simple geometric heuristics. This type of discretized geometric reasoning "distills" out the qualitative features of a given state-space portrait, allowing the representation of and reasoning about these features to proceed at a much higher (and cheaper) abstraction level[4].

Raising the granularity level of the analysis of individual sensor data sets, however, is only a very small part of the power of qualitative reasoning about phase portraits. Dynamical systems can be extremely complicated. Attempting to understand one by analyzing a single behavior instance—e.g., system evolution from *one* initial condition at *one* parameter value, like the limit cycle shown in Figure 1(a)—is generally inadequate. Rather, one must vary a system's inputs and control parameters and study the change in the response. Even in one-parameter systems, however, this procedure can be difficult[2]; as the parameter is varied, the behavior may vary smoothly in some ranges and then change abruptly ("bifurcate") at critical parameter values. A thorough representation of this behavior, then, requires a "stack" of state-space portraits: at least one for each interesting and distinct range of parameter values. Constructing such a stack requires automatic recognition of the boundaries between behavioral regimes, and the cell dynamics representation makes this very easy, as described in conjunction with Figure 1. Specifically, it allows a set of geometrically different and yet qualitatively similar trajectories—an "equivalence class" with respect to some important dynamical property—to be classified as a single coherent group of state-space portraits.

Similar kinds of problems arise in the hybrid systems literature[12]. Hybrid modeling techniques describe continuous nonlinear behavior using an ontology of piecewise-continuous regimes and discrete inter-regime transitions[19]. In this representation, if a control parameter is changed or a state variable moves into a prescribed state-space region, a *transition function* moves or "jumps" the hybrid model into that new operating regime and simultaneously invokes the appropriate governing equations. If one attempts to use this representation to capture the behavior of a nonlinear dynamical system, however, the requirement that different operating regimes occupy physically distinct state-space regions poses some serious problems. The same state-space region may exhibit radically different behaviors for different control parameter values, and the simple hybrid system representation cannot handle this.

Consider, for example, a driven pendulum model described by the ODE

$$\ddot{\theta}(t) + \frac{\beta}{m}\dot{\theta}(t) + \frac{g}{l}\sin\theta(t) = \frac{\gamma}{ml}\sin\alpha t$$

with mass (m), arm length (l), gravity constant (g), damping factor (β), drive amplitude (γ) and drive frequency (α). m, l, g and β are constants; the state variables of this system are θ and $\omega = \dot{\theta}$. In many experimental setups, the drive amplitude and/or frequency are controllable: these are the "control parameter" inputs of the system. The behavior of this apparently simple device is really quite complicated and interesting. For low drive frequencies, it has a single stable fixed point; as the drive frequency is raised, the attractor undergoes a series of bifurcations between chaotic and periodic behavior[7]. These bifurcations do not, however, necessarily cause the attractor to *move*. That is, the qualitative behavior of the system changes and the operating regime (in state space) does not. Traditional bifurcation analysis of this system would involve constructing phase portraits of the system, like the ones shown in Figure 1, at closely spaced control parameter values across some interesting range. Traditional hybrid representations do not handle this smoothly, as the operating regimes involved are not distinct. If, however, one adds a parameter axis to the state space, most of

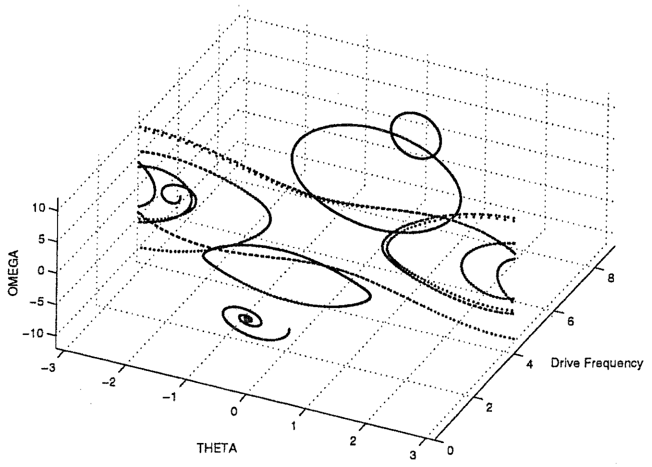

FIGURE 5. A state/parameter (S/P) space portrait of the driven pendulum: a parameterized collection of state-space portraits of the device at various Drive Frequencies. Each (θ, ω) slice of this *S/P-space portrait* is a standard phase portrait at one parameter value. For example, when the drive frequency is zero, the pendulum's attractor is a fixed point; as the frequency is raised, the device goes through various chaotic and periodic regimes and finally settles into a family of limit cycles (the ellipses at drive frequency values of 6 and above). After [9].

these problems vanish. Figure 5 describes the behavior of the driven pendulum in this new *state/parameter space* (S/P space). Each θ, ω slice of this plot is a state-space portrait, and the control parameter varies along the Drive Frequency axis. This idea is similar to the hybrid systems community's idea of forming a cross product of the input space of a system with its output space.

The final step in our development of a good representation for the qualitative bifurcation analysis framework was to combine the state/parameter space idea pictured in Figure 5 with the qualitative abstraction of Hsu's cell dynamics (Section 2.1), to produce the *qualitative state/parameter space*

(QS/P-space) representation. A QS/P-space portrait of the driven pendulum is shown in Figure 6. This representation is similar to the state/parameter-space portrait shown in Figure 5, but it groups similar behaviors into equivalence classes, and then uses those groupings to define the boundaries of qualitatively distinct regions—an effective, useful, application-specific granulation of the behavioral description.

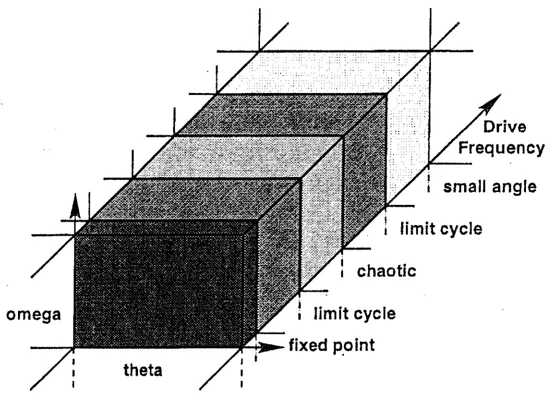

FIGURE 6. A *qualitative state/parameter-space* (QS/P-space) portrait of the driven pendulum. This is a granularization of the state/parameter space portrait in the previous figure; it groups qualitatively similar behaviors into equivalence classes and uses those groupings to define the boundaries of qualitatively distinct regions of state/parameter space. After [9]

This qualitative state/parameter-space representation is an extremely powerful modeling tool. One can use it to identify the individual operating regimes, then create a separate model in each, and perhaps use a finite-state machine to model transitions between them. More importantly, however, the QS/P-space representation lets the model builder leverage the knowledge that its regions—e.g., the five slabs in Figure 6—all describe the behavior of the *same system*, at different parameter values. This is exactly the type of high-level, granular knowledge needed to plan how to learn more about a system by changing its inputs and observing the results. The remainder of this section expands upon these ideas, describing how the QS/P-space representation assists in the automation of input/output modeling of dynamical systems.

3.2 QBA:Reasoning about Granular Knowledge

Reasoning about actuators is much more difficult than reasoning about sensors. The problem lies in the inherent difference between passive and active modeling. It is easy to recognize damped oscillations in sensor data without knowing anything about the system or the sensor, but using an actuator requires a lot of knowledge about both. Different actuators have different characteristics (range, resolution, response time, etc.); consider the difference between the time constants involved in turning off a burner on a gas versus an electric stove, and what happens if the cook is unaware of this difference. The effect of an actuator on a system also depends intimately on how the two are connected. For example, a DC motor may be viewed as a voltage-to-RPM converter with a linear range and a saturation limit. How that motor affects a driven pendulum depends not only on those characteristics, but also on the linkage between the two devices, (e.g., a direct rotary drive configuration versus a slot/cam-follower setup). Planning and executing experiments successfully requires modeling these kinds of properties and effects. One way to do so is to use another granular computing abstraction such as bond graphs[15]. Exploiting the bond graph's two-port nature allows the effects of an actuator to be incorporated into the model quite naturally, via additional modeling components. Since actuators themselves can be complex nonlinear systems, actuators should be modeled initially off-line; that knowledge can then be used as an invariant (that is, regime-independent) part of the actuator/system model.

Qualitative bifurcation analysis requires interleaved input and output reasoning, in which sensors and actuators are used to probe the system at a variety of control parameter values to find interesting behaviors and identify boundaries between different regimes. The QBA process simply scans the range of an actuator at predefined intervals, classifying the results as described above; it then zeroes in on the bifurcations using simple binary search. These bifurcations are the dividing lines between regimes in the QS/P-space portrait of the system, and the qualitative classifications are the labels for the regimes. The results of the QBA process are twofold: a QS/P-space representation of the system dynamics—like the one shown in Figure 6—and a set of qualitative observations similar to those a human engineer would make about the system, such as "When the control parameter ρ is in the range [1.2, 5.6], the state variable x_1 oscillates." The granular information captured in the QS/P-space portrait is useful well beyond the input-output phase of the model-building process; it can also streamline the generate phase, for instance, since a model that is valid in one regime is often also valid in other regimes that have the same qualitative behavior. And even when the qualitative behavior is different, continuity suggests that a neighboring regime's model is a very good starting point.

4 Related Work

A number of researchers have combined numerical techniques with ideas from symbolic computation and artificial intelligence to create granular computational tools for scientists and engineers. One such class of tools autonomously plans, executes, and interprets simulation experiments from high-level specifications of physical models. Abelson's *Bifurcation Interpreter*, for instance, autonomously explores the steady-state orbits of one-parameter families of periodically driven oscillators; it automatically generates a bifurcation diagram and a text description of the findings[2]. A related class of tools addresses the problem of automated phase-portrait analysis, combining ideas from dynamical systems, discrete mathematics, and artificial intelligence to generate qualitative descriptions of different kinds of systems. Bradley's *Perfect Moment* explores a system's state space with particular attention to chaotic features, then uses that information to design nonlinear controllers[3,6]. Yip's KAM extracts useful, high-level information about the phase portraits of Hamiltonian systems by combining computer vision techniques with sophisticated mathematical invariants[24]. Zhao's *Phase Space Navigator* analyzes phase portraits, producing a detailed description of the system dynamics—equilibrium points, boundaries of stability regions, etc.—using a granular analysis tool called the *flow pipe*[25]. The work described in this chapter, which is part of the PRET project[4,5,8,22] is similar to these tools in that it combines traditional numerical computation techniques with symbolic artificial intelligence; it even uses some similar analysis tools (e.g., phase-portrait analysis).

5 Summary

The goal of input-output modeling is to apply a test input to a system, analyze the results, and learn something useful from the cause/effect pair. Automating this analysis procedure is not only important from an engineering standpoint, but also hard and interesting from an artificial intelligence standpoint. In particular, planning, executing, and interpreting experiments requires some fairly difficult reasoning about what experiments are useful and possible, and information granulation plays an extremely important role in this process. The approach described in this chapter uses a hybrid representation termed the *state/parameter (S/P)* space, which granulates information about the behavior of a complex system and the effects of the control parameter upon the behavior. This information then undergoes a second level of granulation—termed qualitative bifurcation analysis—wherein the S/P space is decomposed into discrete regions, each associated with an equivalence class of dynamical behaviors, derived qualitatively using geometric reasoning. In this representation, each trajectory is effectively equivalent, in a well-known sense, to all the other trajectories in the same region, which allows the model builder to describe the behavior of a multiregime system in a significantly simpler way, which results in ease of analysis—and great computational savings.

Acknowledgments. Joe Iwanski and Reinhard Stolle contributed code, and/or ideas to this paper.

6 References

[1] H. Abarbanel. *Analysis of Observed Chaotic Data.* Springer, 1995.

[2] H. Abelson. The Bifurcation Interpreter: A step towards the automatic analysis of dynamical systems. *International Journal of Computers and Mathematics with Applications*, 20:13, 1990.

[3] E. Bradley. Autonomous exploration and control of chaotic systems. *Cybernetics and Systems*, 26:299–319, 1995.

[4] E. Bradley and M. Easley. Reasoning about sensor data for automated system identification. *Intelligent Data Analysis*, 2(2):123–138, 1998.

[5] E. Bradley, M. Easley, and R. Stolle. Reasoning about nonlinear system identification. Technical Report CU-CS-894-99, University of Colorado at Boulder, 2000. In review for *Artificial Intelligence.*

[6] E. Bradley and F. Zhao. Phase space control system design. *IEEE Control Systems Magazine*, 13:39–46, 1993.

[7] D. D'Humieres, M. Beasley, B. Huberman, and A. Libchaber. Chaotic states and routes to chaos in the forced pendulum. *Physical Review A*, 26:3483–3496, 1982.

[8] M. Easley and E. Bradley. Generalized physical networks for model building. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-99)*, 1999.

[9] M. Easley and E. Bradley. Reasoning about input-output modeling of dynamical systems. In *Proceedings of the Third International Symposium on Intelligent Data Analysis (IDA-99)*, LNCS 1642, pages 343–355. Springer, 1999. Amsterdam, The Netherlands.

[10] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3:209–226, 1977.

[11] B.-L. Hao. Symbolic dynamics and characterization of complexity. *Physica D*, 51:161–176, 1991.

[12] T. Henzinger and S. Sastry, editors. *Hybrid Systems: Computation and Control, Proceedings of the First International Workshop, HSCC'98*, LNCS 1386. Springer, 1998.

[13] C. Hsu. A theory of cell-to-cell mapping dynamical systems. *Journal of Applied Mechanics*, 47:931–939, 1980.

[14] C. Hsu. *Cell-to-Cell Mapping.* Springer-Verlag, New York, 1987.

[15] D. Karnopp, D. Margolis, and R. Rosenberg. *System Dynamics: A Unified Approach.* Wiley, New York, second edition, 1990.

[16] M. B. Kennel, R. Brown, and H. D. I. Abarbanel. Determining minimum embedding dimension using a geometrical construction. *Physical Review A*, 45:3403–3411, 1992.

[17] D. Lind and B. Marcus. *Symbolic Dynamics and Coding.* Cambridge University Press, Cambridge, 1995.

[18] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20:130–141, 1963.

[19] P. J. Mosterman, F. Zhao, and G. Biswas. An ontology for transitions in physical dynamic systems. In *Proceedings of the AAAI-98*, pages 219–223, 1998.

[20] N. Packard, J. Crutchfield, J. Farmer, and R. Shaw. Geometry from a time series. *Physical Review Letters*, 45:712, 1980.

[21] F. J. Pineda and J. C. Sommerer. Estimating generalized dimensions and choosing time delays: A fast algorithm. In *Time Series Prediction: Forecasting the Future and Understanding the Past.* Santa Fe Institute, Santa Fe, NM, 1993.

[22] R. Stolle and E. Bradley. Multimodal reasoning for automatic model construction. In *Proceedings of the AAAI-98*, pages 181–188, 1998.

[23] F. Takens. Detecting strange attractors in fluid turbulence. In D. Rand and L.-S. Young, editors, *Dynamical Systems and Turbulence*, pages 366–381. Springer, Berlin, 1981.

[24] K. Yip. *KAM: A System for Intelligently Guiding Numerical Experimentation by Computer.* MIT Press, 1991.

[25] F. Zhao. Computational dynamics: Modeling and visualizing trajectory flows in phase space. *Annals of Mathematics and Artificial Intelligence*, 8:285–300, 1993.